前編

システムの記事

FlexRayの実現に向けた時間駆動型通信リアルタイムOSを開発

――通信プロトコルの実装と自動車メーカの要求の“すり合わせ”

服部博行

筆者らはFlexRay通信の実現に向けて，タイム・トリガ（時間駆動）型通信に対応したリアルタイムOS「Time Triggered Operating System（TT-OS）」を開発した．ここではまず，車載LANに用いられるイベント・ドリブン（事象駆動）型通信とタイム・トリガ型通信の基本を押さえる．また，既存のOSの問題点を洗い出し，新しくTT-OSが開発されるまでの経緯を説明する．（編集部）

自動車における電子制御システムは，ECU（electronic control unit）で制御されています．このECUは，通常，組み込み型のマイクロプロセッサと各種デバイスからなるハードウェア，およびこのハードウェアを制御するためのソフトウェアから構成されています．現在の自動車制御では，このようなECUを複数接続し，相互に情報を共有しながら自動車の機能を実現しています．

ECU間の情報の交換には車載LAN（local area network）が用いられますが，1980年代後半～1990年代には，自動車メーカごとに独自の車載LANプロトコルが策定され，使用されていました．この時代の代表的なLANとしては，米国SAE（Society of Automotive Engineers：自動車技術者協会）が認定した「J1850」やトヨタ自動車の「BEAN（Body Electronics Area Network）」などが挙げられます．

現在，車載LANとして標準的に使用されているCAN（controller area network）は，1986年にドイツのRobert Bosch社がSAEに提案した車載LANプロトコルです．2000年以降になると，車載LANを利用する装置や用途も多様化してきました（**図1**）．CANを利用しなくてはならないほど多くの通信を必要としない部位，あるいはとにかく安価に車載LANを構築したい部位などでは，LIN（local interconnect network）というプロトコルが利用され始めました．また近年，パワートレインやシャーシなどの用途では，ネットワーク内で共通のクロックを使用するタイム・トリガ方式のFlexRayが，次世代の車載LANとして注目されています．

1 タイム・トリガとイベント・ドリブン

今後の制御系（とくにパワートレイン系やシャーシ系）車載LANの標準規格としては，FlexRay通信が有力候補と考えられています．例えば**表1**に示すように，車載LANは四つの段階に分類して考えることができます．FlexRay通

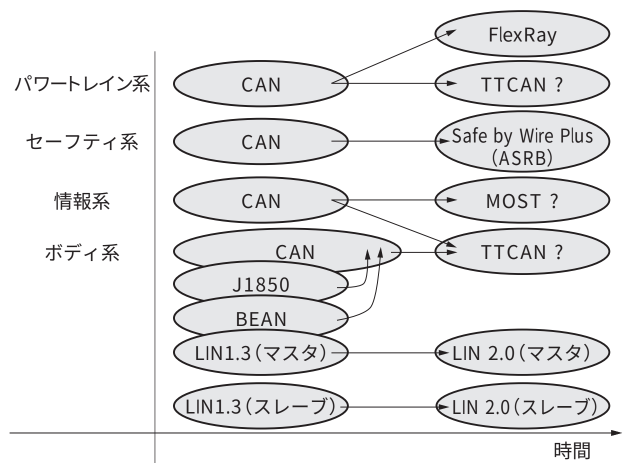


図1　車載LANの進化[(1)]
用途に応じて車載LANも多様化している．

KeyWord　車載LAN，FlexRay，CAN，イベント・ドリブン，タイム・トリガ，OSEK/VDX，TT-OS，X-by-wire

表1 車載ネットワークの発展段階[10]

名古屋大学の高田広章氏による車載ネットワークの分類．

段　階	内　容
第0段階	● 自動車制御にネットワークが使われていない段階 ● エンジン制御，ブレーキ制御が個別に電子化（コンピュータ化）され，それぞれが独立に動作している段階
第1段階 （現世代ネットワーク，CAN通信など）	● 各制御システムが車載ネットワークを通じて，制御品質を向上させるために有用なデータを交換する段階 ● 各制御システムはおおよそ独立して動作 ● 万一，ネットワーク障害が生じた場合，制御品質が下がりはしても，致命的な状況にはならない ● 時間制約は比較的緩やか
第2段階 （現在，この段階の実用化が始まったところ）	● 自動車制御ネットワークを通じて，ミッション・クリティカルでないサービスを提供する段階 ● 個々の制御システムがネットワークを通じて協調動作することで新しいサービスを提供 ● ネットワークに障害が生じると，そのサービスは機能しなくなるものの，自動車の基本機能は保たれる．例えば，安定走行補助機能（トヨタ自動車のVDIM：Vehicle Dynamics Integrated Managementなど），車線維持システムなど
第3段階 （FlexRay通信はこの段階を目ざしている）	● 自動車制御ネットワークを通じてミッション・クリティカルなサービスを提供する段階 ● ネットワークに障害が生じると，自動車の基本機能が損なわれる（X-by-wireシステムなど）

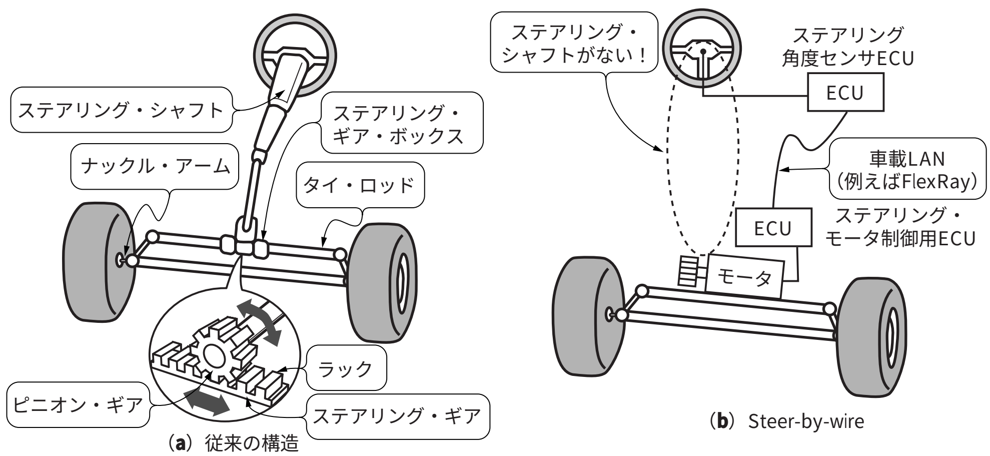


図2 Steer-by-wireの例[9]

（b）のSteer-by-wireでは，ユーザ・インターフェースであるハンドルとステアリング部が機械的につながっていない．つながっているのは，ネットワーク通信路，いわゆる“wire”のみ．

信はこの第3段階を目ざしている車載LANです．この段階では，ネットワークに障害が生じると自動車の基本機能が損なわれるX-by-wireシステムなどの利用を想定しているため，今まで以上の高い信頼性と応答性が求められます（p.118のコラム「なぜ“X-by-wire”なの？」を参照）．

X-by-wireシステムは，従来，機械的に接続されていた部品間を制御ネットワークで接続するシステム全般を示します．代表的なものに，Brake-by-wire，Throttle-by-wire，Steer-by-wireなどがあります．

● CANの応答性ではX-by-wireは実現できない？

図2（a）に示すように，従来のステアリングはハンドルとステアリング部がパイプのような機械部品と歯車で接続されています．そのため，万が一，ステアリングの補助装置（パワー・ステアリングなど）が故障した場合，機能の一部は使えなくなるものの（例えば，ステアリングが重くなるなど），最低限の機能（ハンドル動作に合わせてステアリングは動作するなど）は保障されています．

Steer-by-wireの場合，通信機能障害などでハンドル部の舵角を読み取るためのECUとステアリング部の稼働処理を行うECUが通信できなくなると，ステアリング動作そのものが機能しなくなり，車は曲がれなくなります（**図2（b）**）．Steer-by-wireなどは従来の車載LANより高信頼性で，冗長性のある通信システムでなくてはなりません．

これらの機能は，従来のCANベースの通信プロトコルでは実現が難しいと考えられています．その理由として，CAN通信はイベント・ドリブン方式の通信であり，送信を行うタイミングは多くの場合，アプリケーションにとってつごうの良いタイミングになります．さらに，ネットワーク上のECUは，それぞれ個別の時間（クロック）で実行されており，互いの処理時間を意識していません．そのため，各ECUの送信タイミングはネットワーク上で衝突し[注1]，送信優先順位の低いデータは送信が遅れてしまう可能性もあります（**図3**）．つまり，「送信したい情報が一定時間内に

注1：実際には複数のノードから同時にデータ送信された場合，CANではデータID部の送信時のアービトレーション（調停）により，いずれかのノードが送信権を獲得する．アービトレーションで優先順位が低いと判断されたノードは，直ちに送信を止めて受信動作に移行する．その後，メッセージの送信が完了した後，送信できなかったノードは再送信する．

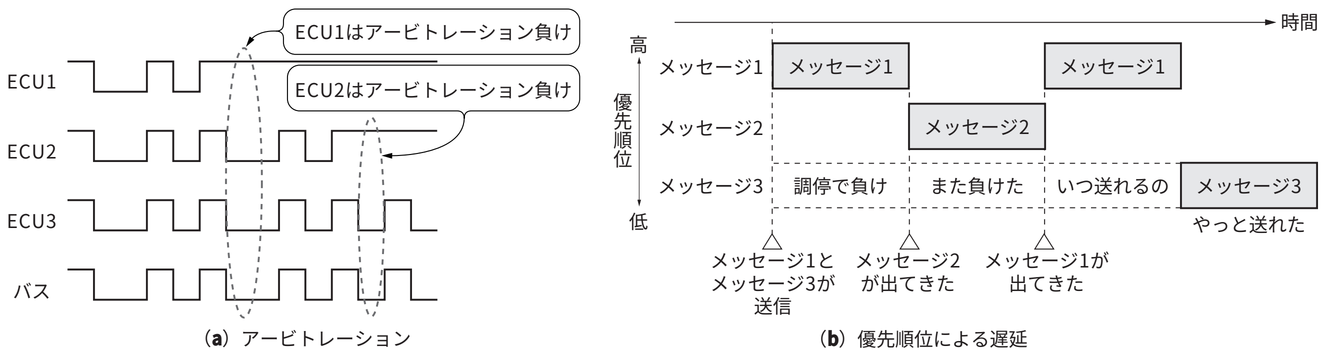


図3　CANの通信方式

(a)にCANのアービトレーションの例を示す．アービトレーションとは，一つのリソースを共有するときに調停して利用すること．CANのアービトレーションはメッセージ先頭のアービトレーション・フィールド（メッセージID）によって行われる．(b)は優先順位による送信の遅延を表している．送信スケジュールが決まっていれば，優先順位の低いメッセージも送信できるのだが…．

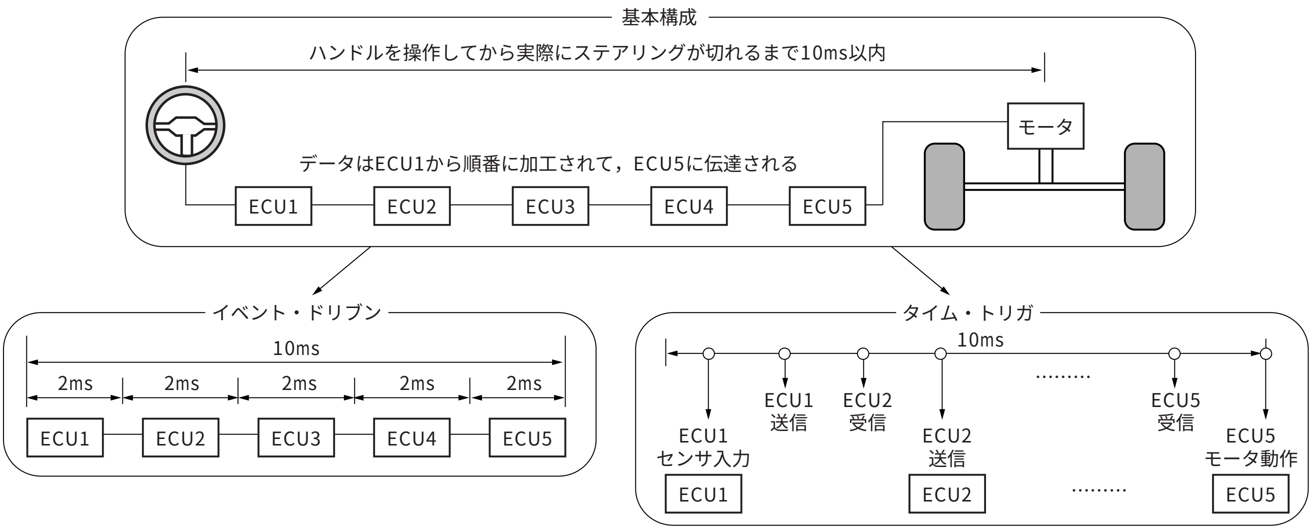


図4　センサ入力から動作までの時間保証

図のような基本構成を持つステアリング動作について考える．イベント・ドリブンの場合，各ECUは2ms以内にデータを入出力する必要がある．最悪のケースを考えると，各ECUは1msで処理するのが望ましく，そうすると高速に処理を行えるプロセッサが必要となる．一方，タイム・トリガでは入力と送信のタイミングさえ正しければよく，残りの処理は比較的ゆっくり制御できる（高速処理のプロセッサは不要）．タイム・トリガのほうが時間保証が容易．

送信相手に到達する保証がない」，言い換えれば「データ送信のリアルタイム性を保証できない」ということになります（**図4**）．このことは，即時応答が求められる制御にはCAN通信は向かない場合があることを意味しています（ブレーキやステアリングで応答が遅れると，運転者は違和感や危険な状況に陥る場合がある）．また，情報をバケツ・リレーのようにECUに伝達する場合の効率もタイム・トリガと比較して良いとは言えません．

● イベント・ドリブンとタイム・トリガの違い

イベント・ドリブン型は，なんらかのイベントが発生したときにそのイベントに対応した処理を行う方式です．例えば，「ボタンを押したら，ブザーが鳴る（あるいはランプが点灯する，など）」といった感じです．処理の要求がなければ処理を休止することができます．この方式はリアルタイムOSともなじみが深く，従来の制御では一般的です．

これに対して，タイム・トリガ型の処理要求は，外部からのイベントではなく，時間的な要因[注2]です．

車載通信で用いられるイベント・ドリブン通信とタイム・トリガ通信という用語は，ネットワーク単位で考えることが多く，本稿における説明もこれに従います．すなわち，現在の車載ネットワークは，通常，複数のノード（ネットワークにつながったECUのこと）で構成されています．各ノードがそれぞれの判断で，自ノードのつごうの良いタ

注2：この場合の時間は，ECUが個別に管理している時間ではなく，もっと広い世界（例えばネットワーク上の時間など）で管理された時間を示す．

イミングで送受信を行う場合，これをイベント・ドリブン型の通信と呼びます(**図5(a)**)．一方，ネットワーク全体ですべてのノードが共通で利用する時間(グローバル時間)を基準として，その時間に同期して決められたタイミングでデータの送受信を行う場合，これをタイム・トリガ型の通信と呼んでいます(**図5(b)**)．

タイム・トリガ方式を用いた通信を実現するためには，各ノードはネットワークが管理している時間に同期して動作しなければなりません．すなわち，ネットワーク内でグローバル時間を管理し，かつ各ノードがそのグローバル時間に同期するための機構が必要となります．

● FlexRayはタイム・トリガ方式を採用

FlexRayのプロトコルはタイム・トリガ方式の通信を前提としています．タイム・トリガ方式を採用することにより，CAN通信で問題となっていた送信タイミングの衝突や低優先度メッセージの送信タイミングの遅延，データ到達時間の保証などが改善されます．

FlexRay通信はミッション・クリティカルなサービス(具体的にはX-by-wire)を提供することを目的としているため，高い信頼性が必要となります．高い信頼性を確保するため，FlexRayの物理層は二重化(チャネルA，チャネルB)されており，たとえどちらかのチャネルに障害が発生したとしても通信が継続されるように規定されています．また，バス・ガーディアン(BG)と呼ばれるデバイスで自ノードのデータ送信タイミングをつねに監視し，不当なタイミングでメッセージ送信を検出した場合，即座に送信を停止させます[1]．このようにFlexRayでは，ノードが異常になったとしてもネットワーク全体に影響を与えないくふうが施されており，デバイス・レベル(バス・ドライバ，バス・ガーディアン，マイクロプロセッサ)で高い信頼性を確保しています．

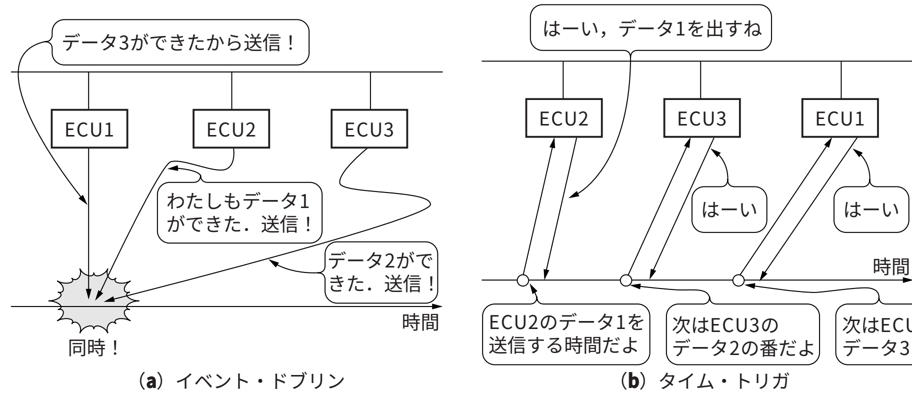


図5 タイム・トリガ通信とイベント・ドリブン通信
(a)のイベント・ドリブン通信では，タイミングは各ノード(ECU)のつごうなので，送信するタイミングがぶつかる(衝突する)かもしれない．一方，(b)のタイム・トリガ通信では，送信タイミングはネットワーク上で定められているため，むだのない通信が行える．

Column 01　なぜ"X-by-wire"なの？

X-by-wireを実現するには既存の通信プロトコルを利用できません．従来よりも高い信頼性や冗長性を必要とするため，高価で安全要求が厳しい車載LANを利用しなければなりません．それにもかかわらず，なぜ"X-by-wire"が必要とされるのでしょうか？

その理由は，X-by-wireでは機械的な装置や歯車，油圧装置などの重量が重い部品の代わりに通信バスを利用するため，車体重量の削減が期待できる点にあります．車体重量を削減できると，燃費が向上します．

また，従来の機械部品を利用する場合，各装置を機械的に接続しているため，決められた位置に配置する必要がありました．しかし，電気的通信による接続になると，位置の制限がなくなり，車両空間で自由なレイアウトが可能となります．例えば，2005年10月22日～11月6日に開催された「第39回 東京モーターショー」で人目を引いた日産自動車のコンセプト・カー「pivo」は，X-by-wireを用いることで運転席をくるりと反転させることができます(後退走行の必要がなくなる)．

また，自動車に求められている新しい機能を実現するためにも力を発揮します．例えば，自動走行の場合，おそらくECUがステアリングやアクセル，ブレーキを操作するものと考えられます．しかし，従来の機械的な接続だと，それぞれの機器を操作する際にその機器を動作させるための特別な装置が必要となります(ハンドルを自動的に動かすにはモータなどを近くに取り付ける必要がありそうだ)．一方，X-by-wireシステムでは，もともとの構成要素として最終的な動作部分にECUとアクチュエータが備わっています．つまり，特別な装置を追加しなくても，自動走行を管理するECUから車載LAN経由でその部分を動作するように指令すれば，自動走行が実現できるのです．

2 タイム・トリガを実現するTT-OS

このようにFlexRay通信に必要不可欠なタイム・トリガ機構を実現するため，筆者らはリアルタイム・カーネル「Time Triggered Operating System（以下TT-OS）」を開発しました．これは，自動車分野で標準になりつつあるOSEK/VDX仕様[注3]のリアルタイム・カーネルに，タイム・トリガ処理を実現するモジュール「Time Triggered Module（以下 TTM）」を追加することで実現しました（つまり，TT-OS＝OSEK OS＋TTMという構成）．基本となるOSEK/VDX仕様のリアルタイム・カーネルには，筆者らが開発し，TOPPERSプロジェクト[注4]が公開している「TOPPERS/OSEKカーネル」を利用しています．

OSEK/VDXの仕様策定団体は，タイム・トリガ機能を実現するリアルタイム・カーネルとして「OSEK/VDX Time-Triggered Operating System（以下 OSEKtime）」の仕様を公開しています．本来なら，筆者らもこのOSEKtimeに準拠したリアルタイム・カーネルを開発し，公開するべきなのですが，欧州の思想に沿ったOSEKtime仕様はかならずしも日本の自動車メーカの考えかたとは一致しません．そのため，OSEKtimeとは異なった仕様を策定し，開発することになりました．

● OSEKtimeの問題点を洗い出した結果，生まれた

当初，筆者らはFlexRay通信を実現するために必要となるタイム・トリガ機能を実現するために，OSEKtimeの開発を検討しました．その検討過程において，自動車でどのようにFlexRay通信を利用するのか，また現在の車載LANはどのように利用されているのかなど，ヒヤリング調査を自動車メーカなどに行ったところ，OSEKtimeの採用にはいくつかの問題となる点があることがわかりました．

OSEKtimeを用いてアプリケーションを構築する場合，OSEK/VDXが推奨するシステム構成は，メインOSにOSEKtimeを利用してタイム・トリガ処理を行います（**図6**）．非タイム・トリガ処理は，タイム・トリガ処理が実行されないOSEKtimeのアイドル時間を利用して，OSEK OSもしくは非OS環境下のメイン処理を実行するようになっています．

図6を見ると明らかなとおり，OSEK OSはOSEKtimeのサブシステムに相当します．この構成では，OSEK OSと比べてOSEKtimeは絶対的な権限を持っています．この場合，タイム・トリガ処理はOSEKtimeの割り込み処理で実行され，タイム・トリガ処理がシステム全体で優先される構成となります．一方，OSEK OSによりサービスが提供されるアプリケーションは，たとえ割り込み処理であってもOSEKtimeが提供するタイム・トリガ処理より優先順位が低く，後回しにされます（**図7**）．つまり，OSEK OSの管理下にあるアプリケーションについては，応答性やリアルタイム性を保証できなくなります．

このようにOSEKtime環境下では，つねにタイム・トリガ処理が非タイム・トリガ（すなわちイベント・ドリブン型）処理より優先されます．FlexRay通信と制御処理をそれぞれOSEKtime環境に当てはめた場合，FlexRay通信はタイム・トリガ処理が必要なのでOSEKtimeによって実現され，OSEK OSによる制御処理に対してつねに優先されることを意味します．

このことのどこに問題があるのかを，エンジン制御を例にして説明してみます．通常，エンジン制御はクランク軸の回転に同期して処理を実行することが少なくありません．点火などのタイミングを逃すと，失火や排気部位での燃焼が発生し，安全性や環境上の問題となるため，クランク軸に同期した処理は必須であると思われます．このエンジン制御部に見られるイベント・ドリブン型の処理は，一般にOSEK/VDX OSで実現します（こうした従来の制御については，イベント・ドリブン型のリアルタイムOS上で実現された資産がある）．エンジン制御で用いる情報交換の手段としてOSEKtimeを用いたFlexRay通信を利用した場

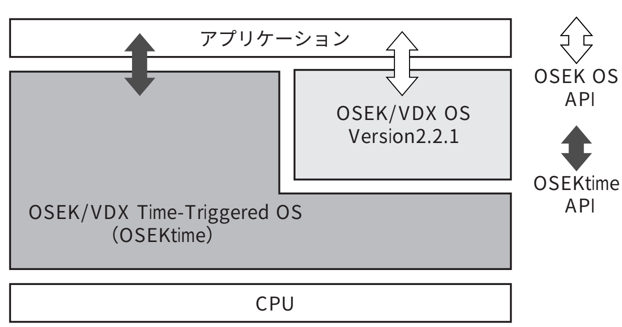
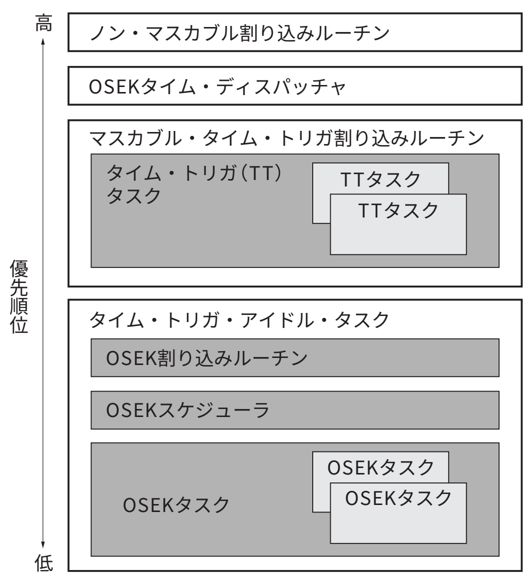


図6 OSEKtimeのソフトウェア構成図
OSEK OSはOSEKtimeのサブシステムに相当する．この構成では，OSEK OSと比べてOSEKtimeは絶対的な権限を持っている．

注3：OSEK/VDXは，ECU間のインターフェースに関する仕様．OS，通信，ネットワーク管理について定義している．本仕様を定めている欧州の自動車メーカを中心とした仕様策定団体の名称でもある．
注4：TOPPERSプロジェクトについてはWebサイト（http://www.toppers.jp/）を参照のこと．

> **Column 02　組み合わせ開発の“OSEKtime”とすり合わせ開発の“TT-OS”**
>
> 最近，自動車関連の業界では「モジュラ（組み合わせ）開発」と「インテグラル（すり合わせ）開発」ということばが流行しています．OSEKtimeとTT-OSの特徴も，このことばで説明できるような気がします．
>
> OSEKtimeは，タイム・トリガを実現するにあたって，タイム・トリガ処理を最優先にしてタイム・トリガ処理部品を容易に持ち込むことができる環境なのだと思います．タイム・トリガを阻害する要因がほかにない環境なので，タイム・トリガ処理の要求から処理完了までの時間的な保証がとりやすい手法であると言えます（あくまでもタイム・トリガ処理の実現を目的とした場合）．
>
> 一方，TT-OSはタイム・トリガ処理とイベント・ドリブン処理を区別することなく，各処理に割り当てられた優先度で実行の順番を判断します．ここに，タイム・トリガ処理とイベント・ドリブン処理のすり合わせ作業が発生します．すり合わせが必要になっても，FlexRay通信を実行しながら同時に制御を実行したいという要望が強いのが日本的かもしれません．



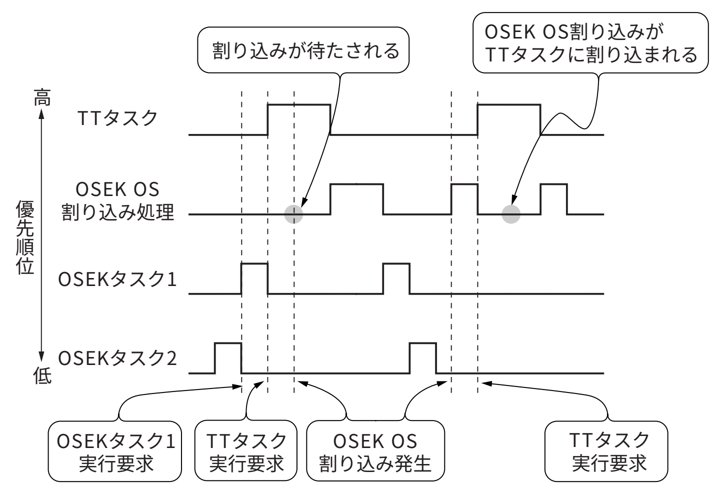


図7　OSEKtimeのプロセッシング・レベル

OSEKtimeでは，非タイム・トリガ（イベント・ドリブン）の制御処理は，つねにタイム・トリガで実行される処理よりも後回しにされる．

合，FlexRayによる通信はOSEKtimeで行われます．このとき，OSEK OS上で稼働するエンジン制御部位は，OSEKtime上で稼働する通信制御のアイドル時間にのみ実行されることとなり，正しくエンジンを制御することが難しくなります．

こうした点を含めて自動車メーカおよび電装部品メーカと意見を交換すると，やはり制御処理を優先したいという意見が多く聞かれました（上掲のコラム「組み合わせ開発の“OSEKtime”とすり合わせ開発の“TT-OS”」を参照）．事実，制御部位のリアルタイム応答性を守らないと，ブレーキ制御などの安全に直接関係する性能を落としかねません．この検討結果を踏まえ，OSEKtime仕様をそのまま実現することは難しいと判断し，この仕様は参考程度にとどめて，新しいタイム・トリガ仕様のTT-OSを開発することにしました．

● TT-OSではイベント駆動処理の応答性の問題を克服

TT-OSの仕様は，タイム・トリガ機能を利用したFlexRay通信を実現することを目的としています．同時に，FlexRay通信とイベント・ドリブン型（非タイム・トリガ型）の制御処理を両立することに注力しています．この方針は，自動車メーカがFlexRay通信の利用を想定している製品を複数選定し，製品に適応する際に必要となる機能および性能を明確にし，分析した結果を反映しています．そのため，TT-OSの機能（特徴）は，OSEKtimeとの比較により明確となります．

まずソフトウェア構成ですが，OSEKtimeと比較するとOSEK OSの位置づけが大きく異なります（**図8**）．また，OSはOSEK OSのみで実現しています．この構成により，イベント・ドリブン型のOSとタイム・トリガ型のOSが干渉することはなくなります．また，同時に複数のOSを用いないので，メモリなどのリソースも少なくてすみますし，

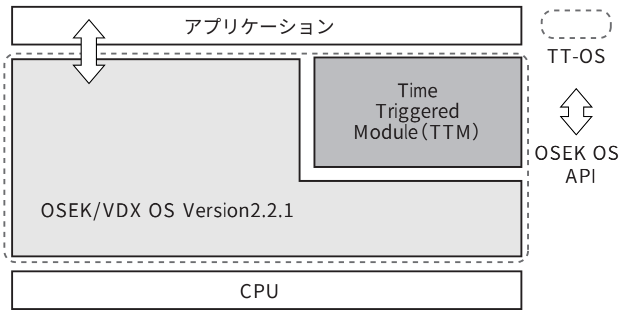


図8　TT-OSのソフトウェア構成図
TT-OSは，OSEK OSのみで実現している．

設計者が複数のOSに精通する必要もありません．

次に大きな違いとして，プロセッシング・レベルを挙げることができます(**図9**)．TT-OSのプロセッシング・レベルは，OSEK OSのプロセッシング・レベルを踏襲しています．すなわちTT-OSでは，イベント・ドリブン型のタスクとタイム・トリガ型のタスクのどちらもOSEK OSの管理下にあり，タスクの起動順序はタスクに与えられている優先順位で決まります．そのため，システム全体で何を優先するべきかを明示的に指定できます．多くのタイム・トリガ型タスクは，イベント・ドリブン型タスクよりも優先度が高くてもよいと思いますが[注5]，通信よりも優先して実行したい処理を指定できるので，前述のエンジン制御などへの適用も可能だと思います．

このような特徴を持ったOSにするため，TT-OSはOSEK OS上にタイム・トリガ処理を行うモジュール(TTM)を追加しています．このTTMは，外部の時間を処理動作のトリガ信号とし，OSEK OSの割り込み処理(OSの管理割り込みで，OSEK OSではISR2と呼ぶ)によって，静的に作成されたスケジュール・テーブルに従って所望する処理の要求を行います(あくまでも要求であり，割り込み内では特別な例を除いて処理を実行しない)．すなわち，TTMではタスクの起動などをOSに要求するだけなので，タスクの起動ポリシなどはOSで一元管理されます．

次回では，TT-OSの仕様について詳しく説明します．

注5：その時間までに処理を行いたいものがタイム・トリガ型タスクに割り当てられると考えられるので，一般にはイベント・ドリブン型タスクより優先度が高くなると思う．

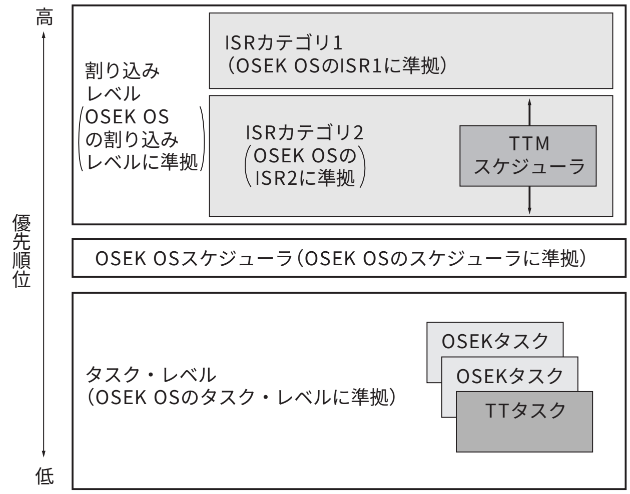


図9　TT-OSのプロセッシング・レベル
TT-OSでは，イベント・ドリブン型のタスクとタイム・トリガ型のタスクのどちらもOSEK OSの管理下としており，タスクの起動順序はタスクに与えられている優先順位で決まる．

参考・引用*文献

(1) 大熊繁監修；車載LANとその応用，トリケップス，2005年6月．
(2) Time Triggered Protocol(TTP/C)：A safety-Critical System Protocol，Howard Curtis and Robert France，EE382C Literature Survey，Oct. 1999.
(3) CIA Webサイト，http://www.can-cia.org/can/ttcan/
(4) OSEK/VDX；OSEK/VDX Time-Triggered Operating System Version 1.0，July 2001.
(5) OSEK/VDX；OSEK/VDX Operating System Version 2.2.1，Jan. 2003.
(6) OSEK/VDX；OSEK/VDX Communication Version 3.0.3，July 2004.
(7) OSEK/VDX；OSEK/VDX Fault ?Tolerant Communication Version 1.0，July 2001.
(8) OSEK/VDX；OIL: OSEK Implementation Language Version-2.5，July 2004.
(9) 原田了；自動車のメカニズム，日本実業出版社，2000年2月．
(10)* 高田広章；オープンソースのOSEK仕様OS，Automotive Technology Days 2004 Autumn，2004年11月1日．

はっとり・ひろゆき
(株)ヴィッツ

<筆者プロフィール>
服部博行．自分の興味のあるものは自分で作りたがるやっかいな性格．日曜大工，ビール作りなどで自分の時間を楽しんでいる．最近は，2人乗りの車でドライブを楽しみ，家族からは白い目で見られているが，本人はまったく気にしていない．